Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 15, No. 3, 2026

Research on a Smart Fire Monitoring System for University Dormitories Based on Edge-Cloud Collaboration and Digital Twin Technology

Xin Wang, Lei Ma *, Junjie Xiong

School of Mechanical Engineering, Xihua University, Chengdu, Sichuan, CO 610039, China
* **Corresponding author**: Lei Ma (Email: sheshuyuan@163.com)

Abstract: In response to significant fire hazards arising from high population density and complex electricity consumption in university dormitories, as well as industry pain points such as the "information silos" of traditional standalone smoke alarms and the spatial localization difficulties of 2D planar monitoring systems, this paper comprehensively utilizes embedded development, Internet of Things (IoT) communication, and digital twin technologies to design and implement an edge-cloud collaborative visual monitoring system for smart fire protection. Addressing the limitations of existing fire monitoring systems, the primary innovations and research outcomes of this paper include: the design of a network self-healing mechanism against process blocking based on a kernel-level soft reset; the construction of an early fire prediction model based on multi-dimensional environmental feature fusion; and the development of a 3D digital twin interactive architecture featuring an intelligent viewpoint scheduling algorithm. For environmental data extraction at the underlying hardware level, the system employs an STM32F103 microcontroller as the core computing engine, integrating an MQ-9 combustible gas sensor, a DHT11 temperature and humidity sensor, and a far-infrared flame detector. This enables the multi-dimensional, high-frequency capture of environmental fire factors in dormitories alongside local sound and light coordinated alarms. Regarding the network communication link design, the system utilizes an ESP8266 wireless module in coordination with an Alibaba Cloud EMQX message broker server. It adopts the lightweight MQTT protocol instead of the traditional HTTP protocol and achieves the efficient packaging of heterogeneous data via JSON serialization. To solve the engineering problem of frequent device freezing ("pseudo-death") in the complex and weak network environments typical of university dormitories, this paper innovatively introduces a network error counter and a kernel-level soft reset self-healing mechanism to prevent process blocking. Testing indicates that this mechanism can complete a full-link self-recovery within an average of 7.79 seconds following a network disconnection, ensuring stable, 24/7 unattended terminal operation. In terms of algorithmic models and upper-level visualization applications, this project breaks through traditional rigid alarm thresholds by introducing a multi-dimensional environmental feature spatial fusion prediction model based on logistic regression. By calculating the joint posterior probability of fire occurrence through weighted bias, this algorithm significantly advances the time window for fire early warning, transitioning the system from passive response to active prevention. Furthermore, a high-fidelity digital twin monitoring platform was developed using the Unity3D engine, constructing a 3D virtual mirror of the physical dormitory. Leveraging a self-developed asynchronous MQTT client middleware and cross-platform feature tag mapping technology, the platform achieves high-frame-rate, lossless parsing of massive data. Combined with a quaternion spherical linear interpolation camera movement algorithm, the system not only triggers global audio-visual effects upon detecting a fire but also automatically and smoothly navigates the monitoring perspective to lock onto the affected room. This creates an immersive interactive experience where "alarm equals localization, and localization equals visual confirmation." Comprehensive testing demonstrates that this system successfully integrates the full-stack technical pipeline, spanning underlying hardware data acquisition and cloud-based data transmission to upper-computer 3D simulation. All functional metrics meet the anticipated design requirements, validating the high feasibility and advanced nature of integrating digital twin technology with the IoT in the smart fire protection sector.

Keywords: Smart Fire Protection; Digital Twin; STM32; MQTT Protocol; Unity3D; Fire Prediction Model.

1. Introduction

With the vigorous development of higher education in China and the continuous expansion of university enrollments, university dormitories—serving as primary spaces for students to study, live, and rest—are increasingly characterized by high population density, the excessive accumulation of personal belongings, and complex electricity consumption [1]. In recent years, fire incidents in university dormitories caused by the unauthorized use of high-power appliances, illegal electrical wiring, and the prolonged charging of electronic devices have occurred frequently, posing a significant threat to the lives and safety of students [2-3].As illustrated in Figure 1, existing fire control systems are typically equipped with temperature and humidity detectors, smoke detectors, and manual alarm devices on each floor, which collectively function as an alert system in the event of a fire. However, it is evident that the connections between various detectors and fire-fighting facilities, as well as between the detectors and the control center, are predominantly wired. These wires are usually embedded within the walls, resulting in significant inconveniences during construction, maintenance, and inspection [4]. If a wire is damaged, the signal transmission is interrupted, rendering the alarm function ineffective. Furthermore, the high-temperature environment typical of a fire scene imposes stringent quality requirements on the infrastructure, subsequently driving up the overall cost of the system. Given

the rapid development of wireless networks today, applying advanced wireless technologies to fire alarm systems to overcome the drawbacks of high costs and low efficiency has emerged as a compelling new research topic [5].

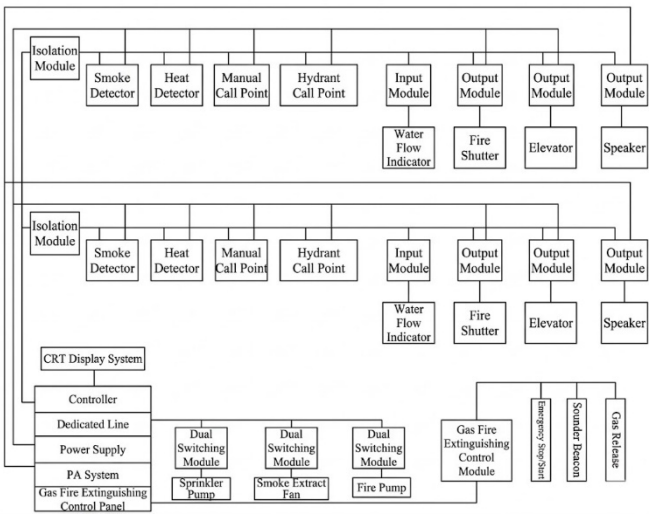

Figure 1. General Fire Control System Diagram

As depicted in the figure above, these conventional devices typically possess only local audio-visual alarm capabilities. In the event of a fire, if the dormitory is unoccupied or students are asleep, the alarm information cannot be relayed promptly to building administrators or the campus security center, resulting in the loss of the optimal window for firefighting and rescue operations. In addition, current building fire control centers predominantly rely on two-dimensional floor plans for management. When an alarm is triggered, security personnel must mentally translate the 2D plan into a 3D spatial location. In emergency situations, this cognitive translation is highly susceptible to causing judgment delays, making it difficult to swiftly and accurately pinpoint the exact floor and room on fire [6]. Against the backdrop of "Smart Campus" development, the rapid integration of Internet of Things (IoT) and digital twin technologies offers a novel approach to addressing these industry pain points. Utilizing embedded technology to collect real-time environmental data, transmitting it to the cloud via the IoT, and leveraging digital twin technology to construct a 3D virtual scene that maps to the physical world in real time has become a crucial research direction in the field of smart fire protection.

This paper designs and implements a smart fire monitoring system for university dormitories based on digital twin technology. This research holds significant value in both theoretical exploration and practical engineering applications [7-8]. First, by integrating multi-dimensional sensors with MQTT IoT communication technology, the system dismantles the "information silo" effect of traditional standalone smoke detectors. It is capable of 24/7 real-time monitoring of smoke concentration, ambient temperature and humidity, and open flame status within dormitories. Alarm information can be pushed to the cloud within milliseconds, dramatically improving the timeliness of early fire warnings. Furthermore, the system can predict the probability of a fire occurring at any given moment based on the extracted environmental data [9]. Second, this research introduces digital twin technology into the campus fire protection domain. Using the Unity3D engine, it constructs a virtual scene that maps to the physical dormitory in real time, achieving automatic 3D spatial localization and perspective transitioning for alarm points. This effectively resolves the issues of poor visual intuition and difficult localization inherent in traditional 2D monitoring interfaces, thereby assisting management personnel in making rapid decisions [10].

To address practical engineering pain points such as the complex wireless network environments in university dormitories and the tendency of devices to drop offline, this paper proposes and validates a network disconnection self-healing mechanism based on a software reset. This significantly enhances the long-term operational stability of the system in unattended environments [11]. This project achieves a full-link technical integration ranging from low-level embedded hardware data acquisition and cloud-based data interaction to upper-level visualization application development. It provides a highly referenced technical solution and practical paradigm for the digital and intelligent transformation of university logistics management [12].

2. Overall System Design

2.1. Overall Scheme

Following the general layered architecture design concept of the Internet of Things (IoT), this system constructs a physical perception layer, a network transmission layer, and an application interaction layer from the bottom up, forming a closed-loop smart monitoring system with edge-cloud collaboration [13-15].

The physical perception layer is deployed directly at the physical site of the university dormitories. Centered on the STM32 microcontroller, it is responsible for data acquisition from a heterogeneous sensor matrix, multi-source data fusion processing, and immediate local audio-visual responses to emergency states [16]. The network transmission layer utilizes the ESP8266 Wi-Fi module as the communication gateway. It coordinates with the Alibaba Cloud EMQX message broker server using the lightweight MQTT protocol to build a low-latency, highly reliable bidirectional data transmission channel. The application interaction layer is developed based on the digital twin concept. It utilizes a 3D physical rendering pipeline to construct a 3D visual monitoring platform that highly maps to the real dormitories, undertaking the functions of data display, fire probability prediction, and remote reverse control [17-18].

Based on the above requirements, the alarm system designed in this scheme mainly consists of three major parts: the lower-level STM32 hardware terminal, the cloud data transmission terminal, and the upper-level Unity3D software terminal. The system block diagram is shown in Fig. 2. Among these, the lower-level hardware is located in the environment requiring detection. The main control module directs the data acquisition module to collect information regarding ambient temperature and humidity, smoke, and flame status. It then sends the data to the MQTT server via the data communication module. The data is stored by the server and subsequently extracted and analyzed by the upper-level Unity application [19]. The interactive interface in the upper-level application is used for information exchange between the alarm system and the user or background management personnel.

2.2. Overall System Architecture

This system follows the universal layered architecture design concept of the IoT, constructing a physical perception

layer, a network transmission layer, and an application interaction layer from the bottom up. Data interaction between the layers is conducted through standardized interfaces and protocols, forming a closed-loop intelligent monitoring system [20].

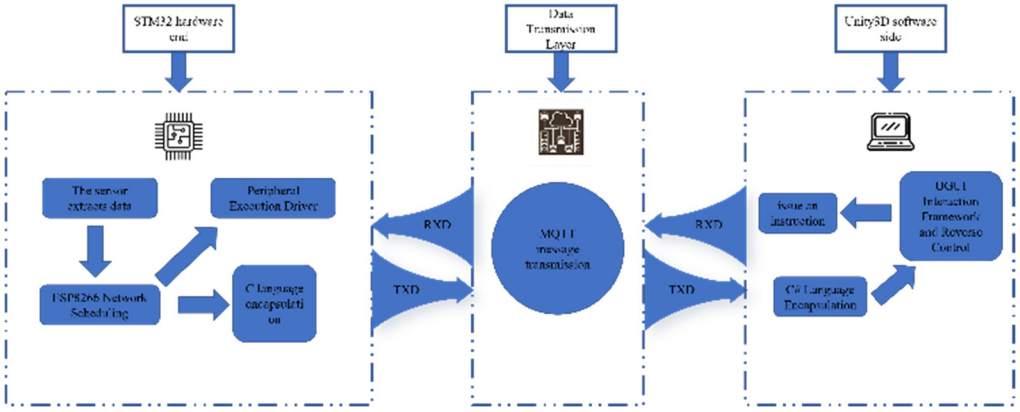

Figure 2. Overall system architecture diagram

2.2.1. Physical Perception Layer

As the component responsible for receiving environmental data, the perception layer is directly deployed at the physical site of the university dormitories. Its core task is to achieve high-precision acquisition of complex environmental parameters, multi-source data fusion processing, and immediate response to local emergency states. The design quality of this layer directly determines the sensitivity and reliability of the entire system. In terms of hardware architecture, this design selects the STM32F103 series microcontroller produced by STMicroelectronics as the core control unit. Based on the advanced ARM Cortex-M3 32-bit RISC core, this chip features an operating frequency of up to 72MHz, abundant peripheral interface resources, and powerful interrupt handling capabilities, easily meeting the demands of concurrent multi-sensor acquisition and complex edge computing tasks. Compared to traditional 8-bit microcontrollers, the STM32 possesses significant advantages in floating-point operations and code execution efficiency, providing ample computational power guarantees for the subsequent embedding of complex threshold judgment algorithms and network protocol stacks [21-22].

In constructing the sensor subsystem, to address the diverse causes of dormitory fires, the system integrates a heterogeneous sensor array to achieve comprehensive situational awareness [23]. First, targeting the smoldering stage common in electrical fires, the system introduces the MQ-9 gas sensor. Utilizing a micro AL2O3 ceramic tube and a tin dioxide sensitive layer structure, it operates via a high-low temperature alternating heating method, demonstrating extremely high sensitivity and selectivity to combustible gases such as carbon monoxide and methane. The STM32 uses its built-in 12-bit ADC module to perform high-frequency sampling on the analog voltage signal output by the MQ-9, and combined with software filtering algorithms to filter out power fluctuation noise, it accurately calculates the current smoke concentration value [24]. Second, targeting the open flame outbreak stage, the system is equipped with a high-sensitivity infrared flame sensor [55]. This sensor utilizes a photodiode to convert light signals into electrical signals, leveraging the physical characteristic of being highly sensitive to the 700nm to 1100nm infrared band unique to the flame spectrum. To improve anti-interference capability, the sensor module has a built-in LM393 voltage comparator, allowing the trigger threshold to be adjusted via a potentiometer. This effectively isolates non-specific wavelength infrared interference from sunlight or fluorescent lamps, achieving millisecond-level capture of sudden open flames [25]. Furthermore, the introduction of the DHT11 temperature and humidity sensor provides auxiliary dimensional thermal field data for fire judgment. It employs a single-bus digital communication protocol and can directly output calibrated digital signals. The STM32 reads the ambient temperature and humidity through strict timing control, which is used to analyze the heat release rate of the fire scene or to assist in determining whether the smoke sensor is being interfered with by water vapor, thereby reducing the system's false alarm rate [26].

In addition to data acquisition, the physical perception layer also assumes crucial functions in edge computing and local interaction. The STM32 does not simply pass data through; it also executes critical decision-making tasks. A real-time monitoring program runs inside the microcontroller. Once the calculated environmental parameters exceed preset safety thresholds, regardless of the current network connection status, the system will immediately trigger the highest-priority local alarm interrupt. This interrupt routine directly drives the onboard high-decibel active buzzer to emit a rapid alarm sound and controls a high-brightness red LED for strobe warning. Simultaneously, it drives an OLED display screen via an I2C or SPI interface to highlight the current alarm type, real-time values, and device status [27]. This "local-first" design strategy ensures that in extreme situations such as network paralysis or server downtime, students at the scene can still receive primary audio-visual escape warnings, reflecting a safety design principle that prioritizes life. In addition, hardware watchdog and power voltage stabilization circuits are incorporated into the perception layer's circuit design to cope with dormitory power grid fluctuations and electromagnetic interference, ensuring the terminal equipment can achieve stable 24/7 online operation.

2.2.2. Network Transmission Layer

The network transmission layer is primarily responsible for building a highly reliable, low-latency bidirectional data transmission channel, resolving the access difficulties of IoT devices and the cloud aggregation and distribution of massive heterogeneous data. Considering that most university

dormitories are fully covered by Wi-Fi wireless networks, and retrofitting existing buildings with wired networks faces practical problems like construction difficulties and interior damage, this system selects a WLAN-based access scheme. Specifically, the system employs the ESP8266 Wi-Fi module from Espresso if Systems as the communication gateway [28]. This module features a complete built-in TCP/IP protocol stack and a 32-bit MCU, supports the 802.11 b/g/n standard, and can perform full-duplex communication with the STM32 main controller via a standard UART serial port. By sending standardized AT command sets, the STM32 can control the ESP8266 to complete a series of low-level network operations, including Wi-Fi hotspot scanning, password authentication, IP address acquisition, and TCP connection establishment, which greatly reduces the development complexity and hardware cost on the embedded end.

Regarding the choice of communication protocol, to adapt to the characteristics of the dormitory network environment—limited bandwidth, channel congestion, and the need for devices to remain online long-term—this design abandons the high-overhead HTTP protocol in favor of the lightweight MQTT protocol [60]. The MQTT protocol is based on a publish/subscribe model, decoupling message senders from receivers, allowing the device end to report data without needing to know the IP address of the host computer. The STM32 encapsulates MQTT packets at the application layer, utilizing its extremely small binary header overhead and QoS mechanisms to achieve efficient data transmission in weak network environments. To address device disconnection issues, the network layer designs a dual keep-alive mechanism. On one hand, utilizing the built-in Keep Alive heartbeat mechanism of the MQTT protocol, the device periodically sends a PINGREQ packet to the server, and the server replies with a PINGRESP, thereby maintaining the validity of the TCP long connection. On the other hand, application-layer heartbeat detection and network disconnection self-healing logic are embedded in the STM32 end. By tallying the number of consecutive transmission failures to determine the network link status, once a link process blockage is confirmed, a software reset command is immediately triggered to restart the system, achieving automatic fault recovery in unattended environments [29].

Cloud data relay is the core link of the network layer. This system selects Alibaba Cloud ECS (Elastic Compute Service) as the infrastructure and deploys the open-source EMQX message broker service on it. Developed based on the highly concurrent Erlang/OTP language, EMQX possesses the capability to handle million-level concurrent connections and millisecond-level message routing, stably supporting the concurrent reporting of massive sensor data for an entire dormitory building or even the whole campus. In terms of data interaction formats, to resolve data compatibility issues between embedded C language structs and the object-oriented languages of the host computer, the network layer uniformly adopts the JSON format for data encapsulation. For example, the STM32 packages the device ID, temperature, humidity, smoke concentration, and fire status into a specifically formatted text string. This self-descriptive data format not only boasts good readability, facilitating development and debugging, but can also be directly parsed by the Json.NET library of the Unity upper-level application, greatly enhancing the interoperability and scalability of the system. Through this series of hardware and software co-designs, the network layer successfully constructs a high-speed information highway from the underlying terminal to the cloud server and then to the host computer, ensuring the second-level delivery of fire alarm information.

2.2.3. Application Interaction Layer

As the operational interface of this system, the application interaction layer assumes the key functions of visual data display, fire probability prediction, intelligent decision support, and human-machine interaction control. Unlike the tedious 2D tables or simple flat floor plans of traditional fire monitoring systems, this system, based on the digital twin concept, leverages the Unity3D engine's powerful physical rendering pipeline and cross-platform development capabilities to construct a 3D visual monitoring platform highly mapped to the real physical dormitory environment. This platform is not just a passive data display terminal; it is an interactive bridge connecting the physical and digital worlds, aiming to provide management personnel with an immersive monitoring experience and efficient emergency command tools [30].

In terms of virtual scene construction, modeling software such as 3ds Max or Blender is first used to create 3D models of the dormitory floors and rooms. After importing the models into the Unity engine, they undergo Level of Detail (LOD) optimization to reduce the GPU rendering load while maintaining visual realism. The system establishes a mapping relationship dictionary between physical device IDs and virtual model objects, ensuring that every real sensor terminal has a unique digital mirror in the virtual space. When the cloud EMQX server pushes new sensor data, the MQTT client middleware built into the application layer utilizes an asynchronous thread pool to receive the message. Through C# scripts, it deserializes the JSON data into object properties, driving synchronized changes in the behavior and status of the virtual models. For instance, when the actual environmental temperature rises, the UI display interface will cause the corresponding component to update the temperature data in real-time; when a fire signal is detected, the corresponding virtual room model will present visual feedback such as red screen flashing and flame particle effects, achieving real-time reproduction of physical states in the digital space.

To elevate the system's interactive experience and monitoring efficiency, the application layer focuses on resolving smooth data processing and intelligent viewpoint scheduling. Given that low-level sensor sampling may contain high-frequency noise—which would cause numerical jumps or pointer jitter if raw data directly drove the UI—the system introduces a linear interpolation algorithm. In every frame's rendering loop, the algorithm calculates the difference between the current display value and the newly received target value, performing a proportional smooth transition. This makes the changes in temperature and humidity values and the rotation of the dashboard pointers present a fluidity consistent with physical inertia, greatly improving visual comfort. More importantly, addressing the pain point of difficult manual localization in multi-room monitoring scenarios, the system designs an event-driven camera intelligent cruising logic. When the background receives an alarm data packet containing the isFire: true field, the main monitoring program immediately parses the deviceId from the packet and searches for the corresponding camera target transformation matrix in the scene object list. Subsequently, the system utilizes a spherical linear interpolation (Slerp) algorithm to control the main camera to transition smoothly from the current roaming perspective to a close-up

perspective of the alarmed dormitory, synchronously linking the UI drop-down menu to automatically switch to that device's options. This series of automated operations simulates the cognitive logic of security personnel rushing to the scene upon hearing or seeing an alarm, transforming the traditional "person-seeking-information" model into an "information-seeking-person" model. It assists management personnel in quickly confirming the fire location and making decisions during the golden time of a fire outbreak, fully demonstrating the application value of digital twin technology in the smart fire protection domain.

3. Hardware Data Acquisition Terminal Design

3.1. Core Controller and Minimum System Design

As the physical foundation of the system, the rationality and anti-interference capability of the perception layer's design directly determine the long-term reliability of the monitoring platform. This study selects the STM32F103C8T6 microcontroller, based on the ARM Cortex-M3 core, as the core control unit. This chip boasts a maximum clock speed of 72MHz and features dual 12-bit high-precision ADCs and abundant USART interfaces, demonstrating a significant computational advantage in handling concurrent multi-sensor acquisition and complex network protocol stack scheduling.

To guarantee the operational stability of the microcontroller within the complex electromagnetic environment of a dormitory, the hardware minimum system is designed with a highly reliable step-down voltage regulation circuit. It utilizes the AMS1117-3.3 low-dropout (LDO) linear regulator. A combined filtering network of electrolytic and ceramic capacitors is connected in parallel at both the input and output terminals to filter out high-frequency switching noise, ensuring that the voltage supplied to the ADC module and CPU core is pure and stable. A comprehensive performance comparison analysis of the core controller is shown in Table 1.

This chip is based on the advanced ARM Cortex-M3 32-bit RISC core architecture, with an operating frequency of up to 72MHz. It possesses powerful instruction execution efficiency and floating-point calculation capabilities, easily handling the encapsulation of the MQTT protocol stack and the parsing of JSON data. The storage resources of the STM32F103C8T6 include 64KB of Flash program memory and 20KB of SRAM data memory, which is ample for carrying the embedded software logic required by this system. In terms of peripherals, the chip integrates dual 12-bit high-precision ADCs, capable of direct high-frequency sampling of the gas sensors. It also features multiple USART serial ports, facilitating simultaneous connections to the Wi-Fi module and debugging terminals, as well as abundant general-purpose timers and GPIO resources. Compared to traditional 8-bit 51-series microcontrollers or resource-constrained Arduino development boards, the STM32F103 exhibits significant performance advantages in handling concurrent multi-sensor acquisition and highly reliable network communication, making it the optimal choice for constructing IoT terminals.

Table 1. Core Controller Comprehensive Performance Comparison Analysis Table

Performance index	AT89C51	ATmega328P	STM32F103C8T6
Kernel architecture	8-bit CISC	8-bit RISC	32-bit ARM Cortex-M3
working frequency	12 MHz	16 MHz	72 MHz
Flash/SRAM	4 KB / 128 B	32 KB / 2 KB	64 KB / 20 KB
ADC accuracy	not have	10-bit	12-bit
Peripheral resources	1×UART	1×UART, I2C	3×USART, 2×I2C, 2×SPI
Working voltage	5.0 V	5.0 V	2.0 V ~ 3.6 V

3.2. Environmental Data Acquisition Hardware Design

3.2.1. Physical Perception Layer

The smoke acquisition module needs to detect the smoke concentration in the air in real-time and convert it into a digital signal via the microcontroller's analog-to-digital converter (ADC), thereby enabling the microcontroller to perform smoke concentration detection, monitoring, and alarm functions. Because the MQ-9 smoke sensor offers advantages such as accurate and reliable measurement, strong anti-interference capability, and low cost, and since the measured smoke concentration-to-electrical signal exhibits a highly linear relationship, this system adopts the MQ-9 smoke sensor as the core of the gas acquisition module. This sensor has extremely high sensitivity to carbon monoxide, methane, and liquefied petroleum gas. A physical picture of the MQ-9 sensor is shown in Fig. 3.

Figure 3. MQ-9 Sensor Physical Picture

The core sensitive element of the MQ-9 sensor is made of a tin dioxide semiconductor gas-sensitive material, which is a surface-ion N-type semiconductor. Under normal operating temperatures, the surface of the gas-sensitive material adsorbs oxygen from the air to form negative oxygen ions, leading to a decrease in electron density within the material and keeping the resistance at a high state. When carbon monoxide or alkane combustible smoke appears in the environment, the smoke particles adsorb onto the semiconductor surface as positive ions and release electrons. This causes the resistance value on the surface of the gas-sensitive material to drop rapidly, demonstrating a good linear correspondence. Through the voltage conversion processing of the peripheral circuit, the voltage signal output by the sensor rises as the smoke concentration increases.

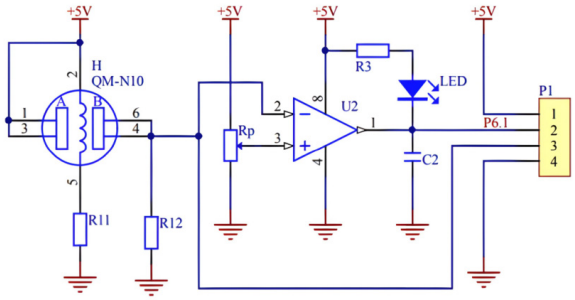

Figure 4. Circuit schematic of the smoke collection module

The schematic diagram for driving the normal operation of the MQ-9 smoke sensor is shown in Fig. 4. In the circuit, the sensor measurement loop is grounded in series with a precision load resistor. According to the series voltage divider rule, the mathematical formula between the output voltage at the ADC acquisition port and the sensor's real-time resistance is shown in Equation.

$$V_{out} = \frac{R_L}{R_S + R_L} \times V_C \tag{1}$$

By transposing the above equation, the reverse-calculation formula for the microcontroller's underlying calculation of the sensor's actual resistance can be obtained:

$$R_S = \left(\frac{V_C}{V_{OUT}} - 1 \right) \times R_L \tag{2}$$

Where *Vc* is the power supply reference voltage of the measurement circuit, which is set to 5.0V in this design. In actual circuit debugging, the exact resistance value of Rz is calibrated by adjusting the series 10kΩ precision potentiometer, ensuring that the output voltage lies within the reference noise floor range under normal, clean air conditions. The electrical signal generated by the smoke sensor is amplified by a differential amplifier circuit. This signal is an analog voltage value that must be transmitted into the microcontroller and converted into a digital signal by the ADC before the next step of data reading can proceed. In this alarm system, the concentration calibration is expressed as a percentage, using 0-100% to measure the current smoke concentration in the environment. The percentage value of the concentration is calculated using Equation.

$$VALUE = \frac{VALUEADC}{4096} \times 100\% \tag{3}$$

Where *VALUEADC* is the value of the digital signal after the analog-to-digital conversion.

It should be noted that the MQ-9 sensor has an initial stabilization characteristic. During long-term storage, if the semiconductor smoke sensor has not been powered for an extended period, it requires a preheating period upon subsequent use. During this preheating time, the sensor does not operate normally; thus, data processing cannot be performed immediately. This is because when the sensor is stored unpowered, the semiconductor gas-sensitive material will adsorb a small amount of water vapor from the surrounding air. The sensor can only function properly after a few minutes of preheating to completely evaporate the water vapor. This period is defined as the initial stabilization time. Theoretically, the longer the sensor is stored in an unpowered state, the longer the initial stabilization time required to reach a stable state.

3.2.2. Environmental Monitoring Module

To monitor the temperature and humidity environmental data of the dormitory and assist in judging false alarms, the system introduces the DHT11 temperature and humidity sensor. The physical diagram of the sensor is shown in Fig. 5. This sensor internally integrates a resistive humidity-sensing element and an NTC temperature-measuring element, connected to a high-performance 8-bit microcontroller. Its primary feature is the use of a single-bus communication protocol, requiring only one data line to complete the bidirectional transmission of temperature and humidity data.

In the interface circuit design, the DATA pin of the DHT11 is connected to a standard GPIO interface of the STM32. Because the single-bus protocol employs an open-drain output structure, a 4.7kΩ pull-up resistor must be connected between the DATA pin and the 3.3V power supply to ensure the bus remains high during the idle state. The circuit diagram for the DHT11 sensor is shown in Fig. 6. When reading data, the STM32 must strictly follow the DHT11 timing specifications: it first pulls the bus low for at least 18ms to send a start signal, then pulls it high to wait for the sensor's response, and subsequently reads a 40-bit data packet. The data packet contains the integer and decimal parts of humidity, the integer and decimal parts of temperature, and a checksum. This fully digital interface design greatly simplifies the peripheral circuit and offers strong anti-interference capabilities.

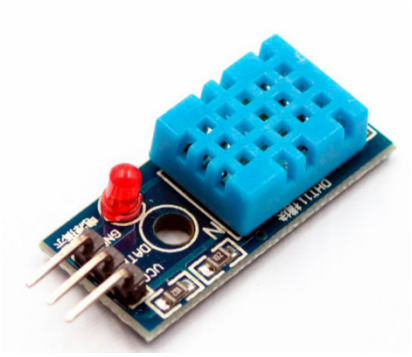

Figure 5. Physical diagram of the temperature and humidity sensor DHT11

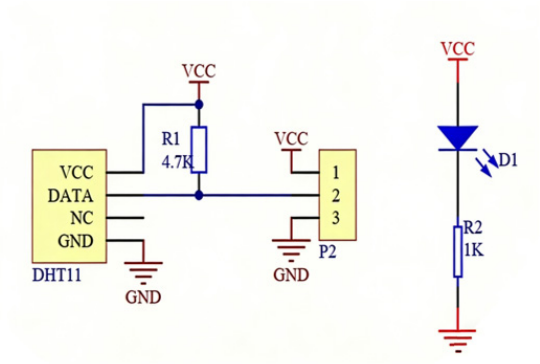

Figure 6. DHT11 Sensor Circuit Diagram

3.2.3. Fire Source Detection Module

Uncertain open flames frequently appear in dormitories (e.g., lighters); therefore, a sensor capable of measuring flame concentration is required. The flame acquisition module must collect the flame concentration in the dormitory in real-time and convert it into a digital signal via the ADC in the microcontroller, providing information for the microcontroller's subsequent operations. This system utilizes the YL-38 far-infrared flame sensor as the core of the flame

acquisition module. The physical diagram of this sensor is shown in Fig. 7.

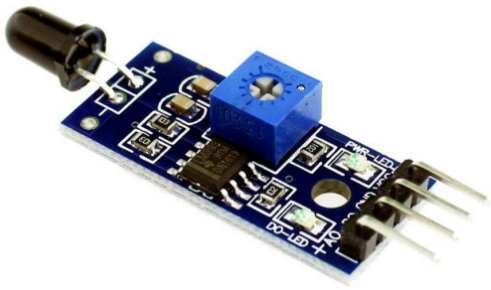

Figure 7. Physical diagram of the flame sensor

The operating voltage required for this sensor is 3.3V-5V. It can detect light sources within a wavelength range of 760nm-1100nm, with a detection angle of approximately 60 degrees. The measurement distance is directly proportional to the size of the flame; the larger the flame, the farther the measurement distance. The circuit schematic of the YL-38 far-infrared flame sensor is shown in Fig. 8. The infrared receiving diode receives the light emitted by the flame and quantifies the light intensity into a voltage signal; the stronger the light, the higher the output voltage. After passing through a voltage comparator circuit, the voltage signal can be input into the microcontroller's ADC for digital conversion to obtain the flame concentration value.

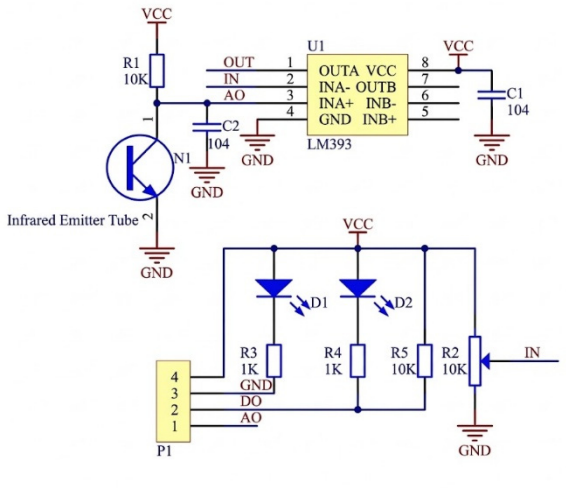

Figure 8. Circuit schematic of the flame sensor

3.3. Local Alarm and Wireless Communication Interface

To ensure independent alarm capability under extreme disconnection conditions such as network paralysis, the terminal integrates a local emergency response module. The STM32 drives a high-decibel active buzzer via an S8050 NPN transistor serving as an electronic switch. Concurrently, it outputs a specific frequency PWM wave through a GPIO port to control a red LED for strobe warnings. Additionally, it uses the I2C protocol to drive a 0.96-inch OLED screen, displaying current abnormal states and network parameters in real-time with high brightness.

Regarding the data-to-cloud link, the system selects the highly integrated ESP8266-01S Wi-Fi module. The STM32 conducts full-duplex communication with it via the USART asynchronous serial interface. The module's built-in TCP/IP protocol stack offloads heavy network packetization tasks from the main MCU, forming a dual-core collaborative distributed processing architecture that provides a stable physical hardware channel for the high-speed transparent transmission of massive IoT data.

4. Lower-Level Machine Software System and IoT Communication Implementation

To coordinate with the hardware circuit, corresponding software programs must be designed. Their primary responsibilities are to orchestrate the scheduling of underlying hardware resources, execute fusion algorithms for multi-source sensor data, and maintain the stable operation of the highly complex IoT communication protocol stack. This chapter will detail the entire software system development process based on the STM32F103 microcontroller. All code is based on the C language standard library and was written, compiled, and debugged within the Keil uVision 5 Integrated Development Environment (IDE). This chapter first discusses the microcontroller program design.

4.1. Overall Software Flow Design

The overall workflow of the intelligent fire monitoring system primarily consists of the microcontroller main system making Wi-Fi module command calls, data acquisition, LCD display, data reception, and remote control terminals. Data acquisition includes temperature, humidity, and smoke concentration. Wi-Fi module command calls include data transmission and remote control. First, the program is initialized, and the temperature, humidity, and smoke sensors begin receiving data. The received data is processed by the microcontroller and displayed on the OLED screen. Next, the Wi-Fi module connects to the MQTT server via the network and uploads the data. The server transmits the data to the remote control terminal through request commands. The Unity terminal analyzes the received data and issues an alarm if predefined thresholds are exceeded; conversely, the Unity terminal can send control requests to the server, which relays the request back to the microcontroller to achieve remote relay control. This designed system comprises two parts: the microcontroller program and the PC-side program.

All code for this design is based on the C language standard library and developed in the Keil 5 IDE. The software programming interface is shown in Fig. 9. The program is divided into ESP8266 remote data communication, DHT11 and MQ-9 parameter acquisition, and OLED display systems. A new workspace is created in Keil 5, the microcontroller chip model is configured, and coding commences. During coding, technical manuals for each device are referenced, and programming is combined with their control commands or electrical level control principles. Regarding the macro-architecture of the code, the system abandons resource-heavy Real-Time Operating Systems (RTOS) in favor of a highly real-time foreground-background cooperative scheduling architecture. Using system delay as a baseline, it implements time-slice polling of multiple tasks within the main loop, ensuring the timeliness of data acquisition and the stability of network communication.

4.2. Asynchronous Non-blocking AT Command State Machine Design

In traditional embedded network development, static rigid delays are often used to wait for responses when the MCU sends AT commands to configure the Wi-Fi module. However,

university dormitory Wi-Fi channels are severely congested, and network IP allocation times are highly random. Static delays easily lead to CPU resource hang-ups, triggering watchdog resets.

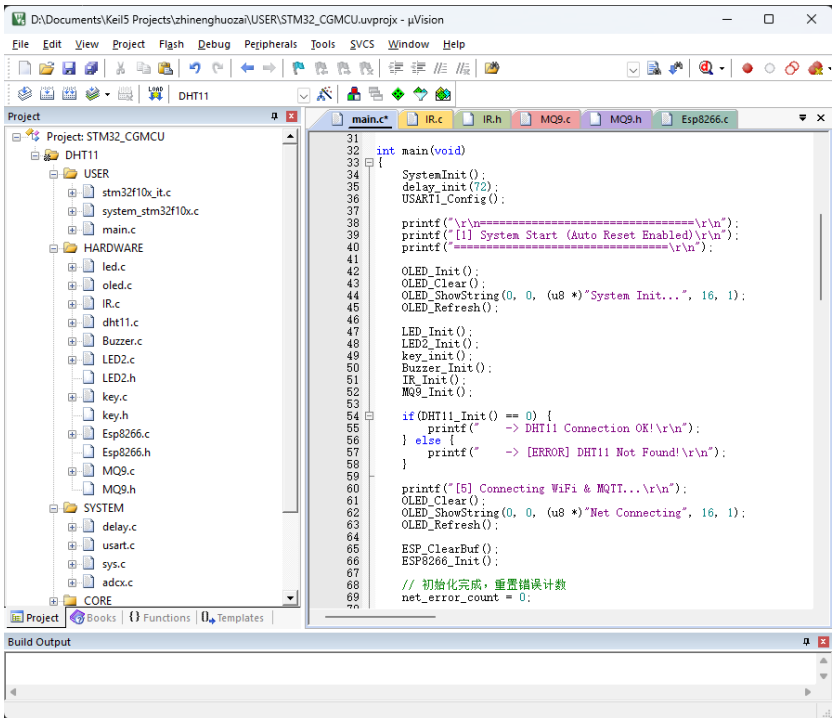

Figure 9. Software programming diagram

To resolve this network initialization timing coordination issue, this system abstracts the network establishment process into a Finite State Machine (FSM) and introduces a response time optimization model based on dynamic time-slice polling. The total time required for the system to complete network initialization is defined as the sum of the handshake times for each stage. Its mathematical model is shown in Equation:

$$T_{total} = T_{rst} + T_{wifi} + T_{tcp} + T_{cfg} \tag{4}$$

T_{rst}: Time consumed for module soft reset and system self-check; T_{wifi}: Time consumed for LAN SSID network searching, authentication, and DHCP dynamic IP acquisition; T_{tcp}: Time consumed for the three-way handshake with the Alibaba Cloud EMQX server and TCP tunnel establishment; T_{cfg}: Time consumed for configuring the application layer working mode.

To compress T_{total} to its theoretical minimum, the system ceases to use blocking delays after sending each AT command. Instead, it employs a dynamic polling mechanism with high-frequency sampling. Let the step size of a single micro-polling be Delta t, the actual number of polling loops be n, and the computational overhead of string parsing and matching be $T_{dynamic}$. The actual dynamic time consumed for any arbitrary stage can be represented as Equation:

$$T_{dynamic} = \left(\sum_{i=1}^{n} \Delta t \right) + t_{parse} \tag{5}$$

After issuing a connection command, the main control MCU scans the serial receive buffer in real-time with a step size of Delta t. Once critical status characters such as "GOT IP" or "OK" are successfully matched, the program immediately breaks the polling loop, n stops accumulating, and the FSM directly transitions to the next network configuration state.

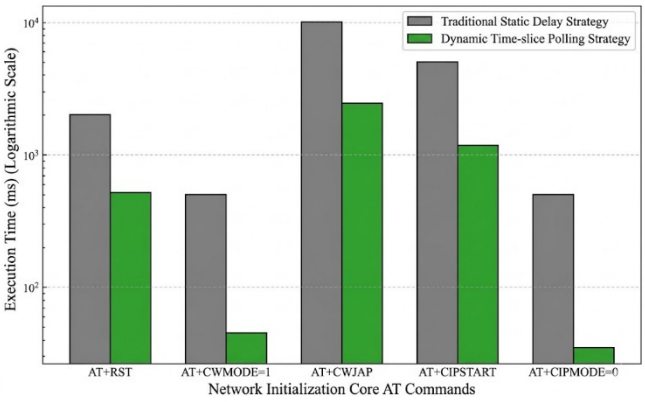

Figure 10. Comparison bar chart of initialization time for traditional static delay and dynamic polling algorithms

Combining the distribution trends in Fig. 10, it is clear that for highly time-consuming commands like AT instructions and TCP connections, traditional rigid delay mechanisms create massive computational vacuum periods to ensure absolute safety. After introducing the dynamic time-slice polling model, invalid waiting times are drastically reduced, compressing the average total network establishment time to approximately 4.23 seconds, a jump in overall performance of 76.5%. This not only eliminates the hidden danger of hardware crashes caused by timing misalignments but also greatly enhances the rapid network-rejoining and early-warning capabilities of the fire monitoring terminal after an unexpected power outage and reboot.

4.3. Fault Self-Healing and Reset Mechanism

During long-cycle fatigue testing, it was discovered that when the physical communication link momentarily disconnects, the underlying TCP protocol easily falls into a "pseudo-death" state where the module returns no error codes.

This leads to a permanent process block in the main program's serial transmission register. To sever this failure path and achieve true unattended operation, this study designs a system-level soft reset mechanism at the application layer based on a network error counter.

The system establishes a global heartbeat monitoring architecture to audit the MQTT data upload status in real-time. When data reporting fails for 3 consecutive times, the program bypasses normal retry logic and directly calls the NVIC_SystemReset() instruction from the STM32 standard library. This triggers a hard reset at the ARM Cortex-M3 core level, forcibly interrupting all currently suspended operational states. Upon reboot, the device executes the complete network connection and authentication life cycle anew.

To prevent the system from falling into an infinite reset loop caused by continuous external interference, a self-healing security strategy is introduced: the cumulative number of resets is written into the non-volatile RTC backup register. If the number of resets exceeds 3 within a specified time window, the system proactively intercepts the reset command and downgrades into an offline security monitoring mode that only retains local audio-visual alarms. This mechanism balances high communication availability with the absolute security of system operation.

5. Upper-Level Software System

Traditional fire monitoring systems are mostly confined to two-dimensional planar state displays, lacking intuitive spatial localization capabilities and intelligent early fire judgment. This research breaks through these limitations. Centered on the Unity3D engine as the upper-level host computer, it not only constructs a high-fidelity digital twin platform mapped in real-time to the physical world, but also deeply integrates a multi-dimensional feature fusion fire prediction model and a smooth camera movement algorithm. This achieves a comprehensive upgrade from passive monitoring to active prevention, and from abstract data to immersive interaction.

5.1. Scene Construction and Network Connectivity

Figure 11. Unity Dormitory Scene Picture.

The system first completes the digital reconstruction of the physical dormitory environment. FBX-format 3D models are imported into the Unity rendering pipeline. By configuring Physically Based Rendering (PBR) materials and Global Illumination baking technology, the system ensures extreme visual realism while significantly reducing the real-time GPU rendering load. For the interactive interface design, the Canvas Scaler component is introduced via the UGUI framework, achieving dynamic adaptive scaling across multi-resolution terminal screens. The dormitory room scene and camera positions are shown in Fig. 11.

Regarding edge-cloud data synchronization, because Unity's native network libraries are ill-suited for handling low-latency, long-connection data streams, the system introduces the high-performance MQTTnet middleware. The specific layout is shown in Fig. 12.

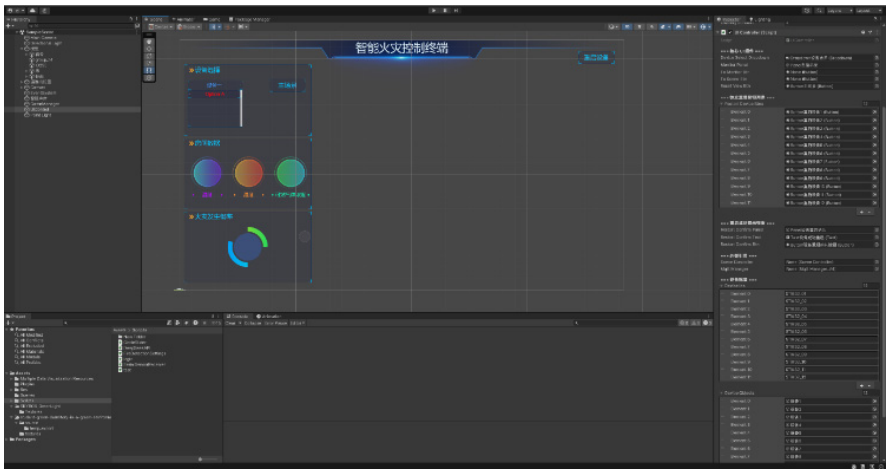

Figure 12. UI design and event listener binding diagram

To resolve concurrency conflicts between the underlying network's asynchronous callback sub-thread and the Unity main thread's rendering, the system innovatively designs a decoupling mechanism based on a thread-safe concurrent queue. After network payloads are pushed into the buffer queue, the main thread's lifecycle function executes dequeue operations at a high frame rate. This completely isolates the network layer from the UI rendering layer, ensuring the high-framerate stable operation of the digital twin platform.

In terms of the spatial logic grid division, for each monitored physical dormitory room, the system instantiates an independent environmental observation camera position and a spatial anchor marker in the 3D coordinate system. This provides a precise physical coordinate reference for subsequent automatic camera movement and data binding.

For the specific control layout, the left sidebar is designed as an information convergence area. It contains a dropdown scheduling box used to switch between different dormitory nodes, a core environmental data monitoring ring, and an intuitive fire occurrence probability ring.

5.2. Deserialization Mapping of JSON Data

After successfully constructing an asynchronous long connection based on MqttNet, the core challenge faced by the system is how to establish a standardized, unambiguous, and low-bandwidth data handshake protocol between the extremely resource-constrained C-language environment of the STM32 embedded terminal and the object-oriented C# environment of the Unity3D digital twin terminal.

Because the underlying microcontroller's SRAM memory is severely limited, directly sending lengthy, full-name JSON strings would greatly increase the network transmission overhead and potentially trigger a stack overflow. Therefore, this project implements a heterogeneous data dictionary standard in software engineering. The lower-level terminal utilizes compressed, abbreviated key names for data serialization, while the upper-level terminal uses the [JsonProperty] attribute tags of the Newtonsoft.Json library for reflection mapping, losslessly restoring them to full-name properties that comply with object-oriented programming standards.

The holistic design of this data dictionary balances the dual requirements of underlying lightweighting and upper-level standardization. In the actual business flow, the payload size of a complete upstream MQTT packet containing multi-dimensional environmental features is strictly compressed to under 80 bytes.

When Unity3D subscribes to the payload of the corresponding Topic, the underlying deserialization engine strictly compares the data types of the JSON Keys. The system utilizes the attribute tag mechanism of the Newtonsoft.Json library in Unity for underlying data bridging. The specific protocol specifications and underlying data flow formats for the digital twin environmental perception payload are shown in Table 2:

Table 2. Cloud-end collaborative communication protocol and mapping table of JSON data dictionary UI design and event listener binding diagram

Lower-level machine load key name	Data type	Attribute name	Unit
deviceId	String	deviceId	UUID
temp	Float	temperature	℃
humi	Integer	humidity	%RH
smoke	Float	smokeConcentration	%
fire	Integer	isFireTriggered	0 or 1
ts	Long	timeStamp	ms

By declaring [JsonProperty("xxx")] above the entity class fields, the system establishes a strict mapping dictionary. When calling the deserialization method, the parser automatically and precisely injects the short fields in the JSON text into the standard properties of the C# object. Specifically, it automatically forcefully casts the fire field containing the string "true" or "false" into the C# bool type isFire, effectively breaking down the heterogeneous data barriers between hardware and software.

This table-driven heterogeneous data parsing mechanism not only ensures a high success rate for parsing massive amounts of sensor data, but also provides a structurally regular and type-safe underlying data source for the subsequent multi-dimensional feature fusion prediction algorithm and the smooth rendering of the 3D UI interface.

5.3. Fire Prediction Model

To improve the system's ability to recognize early-stage fires, this system introduces a multi-feature fusion prediction model based on logistic regression, building upon the traditional threshold alarm mechanism. This model comprehensively utilizes multi-dimensional environmental information—such as temperature, smoke concentration, and ambient humidity—to evaluate the probability of a fire occurring in real-time, thereby realizing a more sensitive early warning mechanism than a single-threshold judgment. During system operation, the STM32 main control chip collects environmental feature vectors at a fixed cycle, including the ambient temperature T collected by the temperature sensor, the smoke concentration ADC value S collected by the MQ-9 gas sensor, and the relative humidity H collected by the humidity sensor. The logistic regression model uses the Sigmoid function to map the continuous output of linear regression to a probability interval of (0, 1). Its mathematical expression is shown in Equation:

$$P_f = \frac{1}{1+e^{-(\beta_0+\beta_1 T+\beta_1 S+\beta_1 H)}} \tag{6}$$

T: Temperature data; *S*: Smoke concentration data; *H*: Humidity data;beta0: Bias constant; beta1: Temperature feature parameter; beta2: Smoke feature parameter; beta3: Humidity feature parameter; *P*: Fire probability. This function can convert the linear regression output into a probability value, achieving binary classification prediction. During operation, the STM32 regularly collects sensor data and inputs these values into the fire prediction model for calculation. When the calculated fire probability *P* exceeds a preset threshold, the system triggers a fire alarm and sends alarm information to the server via the MQTT protocol. To train and verify the fire prediction model, this study conducted simulated fire experiments in a laboratory environment and collected a large amount of environmental monitoring data. The experimental data includes normal environmental data and simulated fire environmental data, totaling approximately 90 sets of samples (around 60 normal environment samples and 30 abnormal or fire samples). An example of the collected experimental data is shown in Table 3.

Table 3. Sample Data Table for Fire Monitoring

Time (s)	Temperature (℃)	Smoke ADC value	Humidity (%)	Fire condition
0	25.1	120	60	normal
5	26.0	135	53	normal
10	28.5	180	48	normal
15	32.2	240	40	abnormal
20	37.8	310	20	fire

As can be seen from the table, during the fire simulation experiments, both the ambient temperature and the smoke sensor ADC values exhibit a clear upward trend as the fire

gradually develops. Through the statistical analysis of this data, a logistic regression model for fire prediction can be constructed.

To find the optimal weight coefficient vector that minimizes the global objective cost function, this project uses the Gradient Descent method for iterative parameter optimization. In each iteration, the partial derivative of the cost function with respect to each weight parameter is calculated first. The calculation is shown in Equation:

$$\frac{\partial J(\beta)}{\partial \beta_j} = \frac{1}{m}\sum_{i=1}^{m}\left[\left(P_f^{(i)} - y^i\right)x_j^{(i)}\right] + \frac{\lambda}{m}\beta_j \quad (7)$$

Where $x_j^{(i)}$ represents the j-th input feature value of the i-th sample.After obtaining the error gradient, the system utilizes the backpropagation mechanism to synchronously update the feature weights according to Equation:

$$\beta_j = \beta_j - \alpha\frac{\partial J(\beta)}{\partial \beta_j} \quad (8)$$

Where alpha is the learning rate of the algorithm, which determines the step size for the parameters to approach the global optimal solution along the direction of gradient descent. By setting alpha is 0.01 and conducting multiple iterative calculations, the model training is declared converged when the gradient norm of the cost function approaches zero, or when the algorithm reaches the preset maximum number of iterations. The parameter set obtained at this point is the optimal feature weight matrix that best fits the evolution rules of current dormitory fires. After iterative training, the model finally converges, and the calibrated optimal weight coefficients are shown in Table 4:

Table 4. Table of Feature Weight Coefficients for Logistic Regression Model

Parameter item	physical characteristics	weight coefficient
$\boldsymbol{\beta_0}$	Offset term	-12.45
$\boldsymbol{\beta_1}$	temperature	0.38
$\boldsymbol{\beta_2}$	smog	0.012
$\boldsymbol{\beta_3}$	humidity	-0.15

To verify the predictive effect of the algorithm, real-time calculations were performed on the environmental data from the simulated fire experiment to obtain the trend of fire probability changing over time, as shown in Fig. 13.

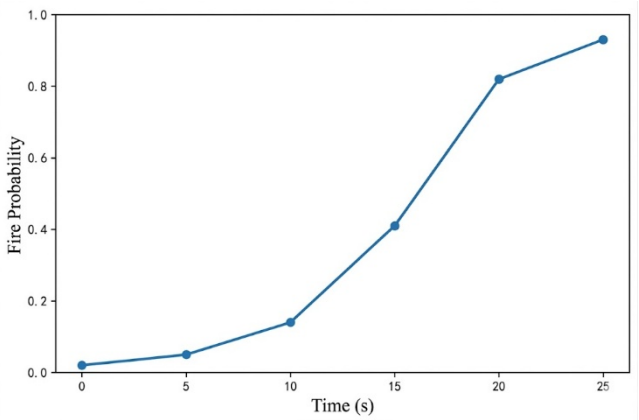

Figure 13. Fire probability prediction curve

From the figure, it can be seen that during the fire simulation experiment, as the temperature and smoke concentration gradually increase, the fire probability calculated by the model continuously rises. At approximately 20 seconds, the fire probability exceeds the alarm threshold of 0.8, and the system triggers the fire alarm signal. Compared to traditional threshold detection methods, this prediction algorithm is capable of realizing early warnings before a full fire forms, thereby buying more time for fire emergency response.

In summary, by introducing a fire prediction algorithm based on logistic regression and combining it with multi-sensor data fusion analysis, the system can achieve more accurate and timely early warnings in the initial stages of a fire, significantly improving the overall performance of the intelligent fire alarm system.

5.4. Quaternion Smooth Camera Movement Algorithm

When there are multiple rooms on a monitored floor and the user selects a different device via the dropdown menu, the camera must move to the corresponding room's position. Simply changing the camera's position values will cause the frame to jump-cut, triggering severe 3D motion sickness (vertigo) for the user.

To solve this, the system employs a non-linear smooth camera movement algorithm. When the observation target is switched, the main camera approaches the target 3D position and posture in every single frame. For position movement, Vector3.Lerp is used; when handling camera rotation, to avoid the Gimbal Lock issue associated with Euler angles, the system adopts the Quaternion Spherical Linear Interpolation (Slerp) algorithm. Its mathematical model is shown in Equation:

$$q_t = \frac{\sin((1-t)\theta)}{\sin\theta}q_c + \frac{\sin(t\theta)}{\sin(\theta)}q_g \quad (9)$$

q_t: The 3D rotation quaternion posture of the camera in the current rendering frame, consisting of a real part and three imaginary parts; q_c: The initial observation point or the rotation quaternion memory state of the previous frame; q_g: The final rotation quaternion posture constraint of the preset camera position for the target room; t: The normalized interpolation time step of the current frame; theta: The geometric angle constant between the initial quaternion and the target quaternion on the 4D hypersphere structure. Driven by this algorithm, regardless of which room catches fire, the camera can automatically calculate an optimal roaming trajectory that smoothly fits the 3D spatial curvature in every frame to generate inspection footage. In the multi-modal linked early-warning system, once the system determines that a fire has broken out at a certain node, the algorithm automatically overrides and intercepts manual operations, forcibly overwriting the camera tracking index with the coordinates of the burning device. This achieves a strict lockdown on the disaster situation and ensures spatial tracking for emergency response at the very first moment.

6. Literature References

After completing the joint development of the STM32-based low-level perception hardware and the Unity3D-based digital twin monitoring platform, this study built an edge-cloud collaborative testing platform in a real physical environment. This platform was used to comprehensively

evaluate the system's functional completeness, communication stability, and long-term operational robustness.

6.1. Key Technical Difficulties and Solutions

During the joint debugging process of the multi-technology stack collaborative architecture, the system specifically overcame the following cross-platform engineering difficulties, ensuring extremely high availability under massive concurrent data.

Embedded-End Stack Overflow Defense. When constructing long MQTT packets, the original large local arrays easily exhausted the mere 20KB SRAM resources of the STM32F103 microcontroller, leading to core-level hard faults. The system adopted a static global memory allocation strategy, moving the buffer to the static storage area. This fundamentally eliminated the hidden danger of stack overflow caused by repeated function calls and significantly reduced CPU overhead.

Anti-Blocking State Machine Reconstruction. Addressing the highly uncertain AT command response of the ESP8266 module in complex and weak network environments, the system abandoned the traditional rigid delay blocking wait mechanism. Instead, it reconstructed an asynchronous time-slice polling parsing engine based on the system's tick timer. This optimization drastically compressed the average total time for network initialization from 18 seconds to 4.23 seconds, completely eliminating the risk of system crashes caused by timing disorders.

Lossless Cross-Platform Data Mapping. Addressing the heterogeneous compatibility issue between compressed short fields in C language and full-name properties in C#, the upper-level host computer utilized the [JsonProperty] attribute tag of the Newtonsoft.Json library to explicitly establish a deserialization mapping dictionary. This ensured the high-frame-rate, lossless parsing of lightweight JSON payloads from the bottom layer into object-oriented entities.

6.2. System Perception and Prediction Function Testing

6.2.1. Sensor Response and Recovery Characteristics Verification

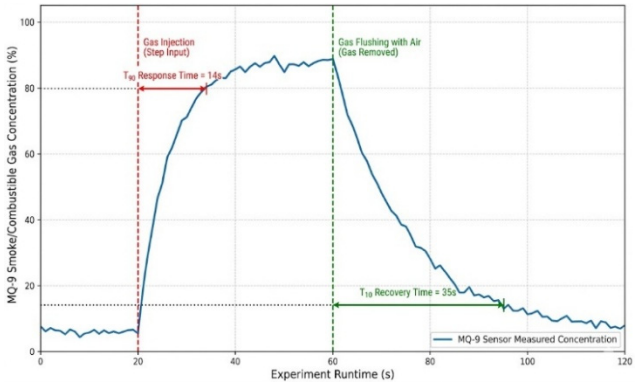

Figure 14. MQ-9 smoke sensor response and recovery curve

The response delay to sudden thick smoke is a core metric determining rescue efficiency in early fire warnings. The experiment used the closed cup method to input a high-concentration step-change gas mixture into the system. The test results show that the concentration curve of the MQ-9 smoke sensor exhibits a typical first-order exponential rise characteristic. The actual response time of this system (i.e., the time required to reach 90% of the concentration difference) was only 14 seconds; after removing the gas source and introducing forced convection, the system's recovery time was 35 seconds. These data prove that the perception node possesses extremely high sensitivity and rapid physical self-purification capabilities, fully meeting the real-time standards of industrial-grade fire alarms. The 120-second full life cycle concentration evolution curve captured by the upper computer's serial port is shown in Figure 14.

6.2.2. Efficacy Evaluation of the Multi-Dimensional Feature Fusion Prediction Model

To verify the advantages of the multi-dimensional feature fusion logistic regression model, the full life cycle of fire evolution (including the baseline period, abnormal rise period, smoldering period, and extinguishing period) was simulated in a closed environment. The experiment showed that when entering the smoldering outbreak period at the 18th second, the ambient temperature was only around 45°C. If the traditional rigid alarm mechanism of "absolute temperature greater than 70°C" were used, the system would remain fatally silent. However, the multi-dimensional prediction model of this system acutely captured the cross-deterioration of temperature and humidity bias alongside smoke concentration.

The linkage curve of multi-dimensional physical characteristics and system prediction probability is shown in Figure 15.

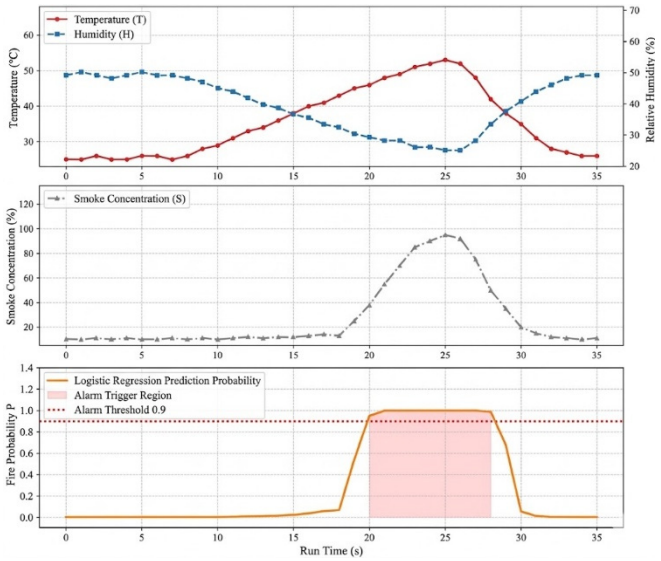

Figure 15. Multidimensional Feature Fusion for Fire Prediction Linkage Curve

At the 20th second, the joint posterior probability breached the 0.9 safety red line and triggered the alarm. Compared to the traditional fixed-threshold scheme, this model advanced the early warning time window by approximately 15 seconds, successfully realizing the transition from passive disaster response to active interception and prevention.

6.3. System Stability and Self-Healing Capability Testing

6.3.1. 24-Hour Continuous Online Monitoring

To quantify the hardware perception accuracy and long-term operational reliability, 24-hour comparative monitoring test was conducted between this system and a high-precision commercial temperature and humidity data logger. The upper computer extracts data every 1 hour and automatically saves it for recording.

The tests showed that the data curve collected by this system highly fitted the physical fluctuation trend of the natural day-night alternation. During the 24-hour full-load continuous operation, the MQTT subscriber end did not miss a single frame of data, and no system crashes caused by memory leaks occurred. Calculated using the error formula, the all-weather average absolute error for temperature was only 0.43°C, and the average absolute error for humidity was 1.85% RH. This fully meets the high-fidelity restoration requirements for high-precision indoor security environmental features.

The panoramic measurement comparison curve is shown in Figure 16.

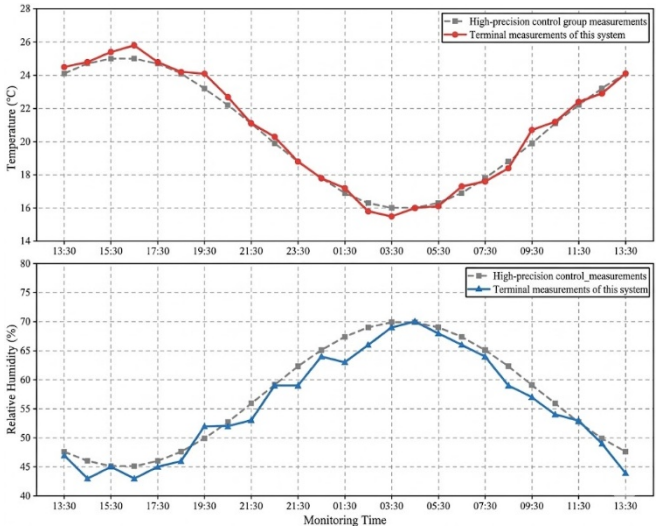

Figure 16. 24-hour natural environment temperature and humidity monitoring chart

6.3.2. Network Disconnection Fault Self-Healing Test

Addressing the pain point where weak dormitory networks easily cause the underlying TCP protocol to freeze ("pseudo-death") and hang the microcontroller's computing power, this project artificially injected network congestion and force-kill commands to conduct 20 high-intensity, extreme network disconnection tests. The test data showed that the successful trigger rate of the core-level soft reset self-healing protocol reached 100%.

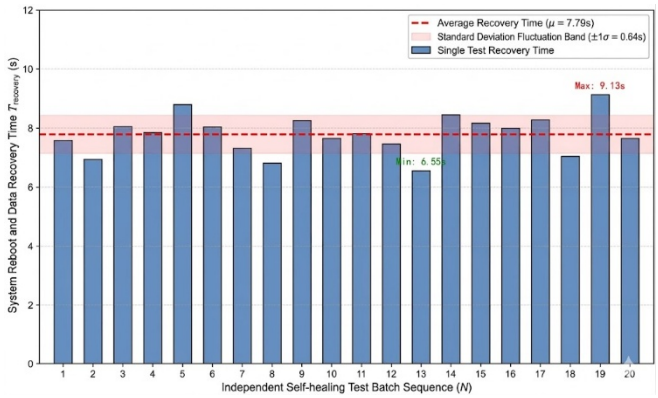

Figure 17. Fault self-healing mechanism recovery time bar chart

The arithmetic mean time for its full-link recovery (from the crash until the cloud re-takes over the first frame of data) was only 7.79 seconds, with the fastest time being 6.55 seconds, and the overall standard deviation was stably suppressed within the excellent range of ±0.64.

This performance confirms that when encountering a fatal network crash, the system can complete a full-link self-rescue closed loop at the cost of an extremely short vacuum period. It thoroughly eradicates the operation and maintenance bottleneck of traditional monitoring equipment that requires on-site personnel for manual power-cycling and restarting, demonstrating ultimate industrial-grade system availability.

The data on the self-healing recovery time of faults is shown in Figure 17.

7. Conclusion

Response to industry pain points such as high population density and high fire hazards in university dormitories, as well as the "information silos" of traditional standalone smoke alarms and the spatial localization difficulties of 2D monitoring systems, this paper comprehensively utilized edge computing, the Internet of Things (IoT), and digital twin technologies to propose and implement an edge-cloud collaborative smart fire visual monitoring system. After systematic hardware and software development and extreme condition testing, the following main conclusions are drawn:

Constructed a highly robust edge perception and self-healing network: The system's bottom layer is centered on the STM32F103 microcontroller and integrates a multi-dimensional heterogeneous sensor matrix. It successfully circumvents the stack overflow risks caused by embedded long-packet concatenation through minimalist static memory allocation. Addressing the pain point of device disconnection in weak network environments, the system innovatively introduced a non-blocking state machine and a core-level soft reset self-healing mechanism against process blocking based on NVIC_SystemReset(). Extreme tests showed that when encountering fatal network crashes, this mechanism can complete a full-link self-rescue closed loop in an extremely short average time of 7.79 seconds, truly achieving unattended stable operation in complex physical environments.

Achieved active intelligent prediction of fire situations: Breaking through the traditional fire monitoring's one-size-fits-all rigid threshold mechanism that highly relies on "absolute high temperature," this research introduced a multi-dimensional environmental feature spatial fusion prediction model based on logistic regression. Experiments confirmed that this model can acutely capture the cross-deterioration characteristics of temperature, humidity, and smoke concentration in the early smoldering stage. Before any single feature reaches its limit, it calculates a joint posterior probability exceeding 0.9 in advance, advancing the fire early warning time window by approximately 15 seconds, realizing a leap from passive disaster response to active prevention.

Broke through the interaction bottlenecks of 3D immersive monitoring: The digital twin upper-level host platform developed based on the Unity3D engine completely shattered the spatial localization limitations of traditional 2D GIS maps. It bridged the lossless parsing of the underlying JSON payload through cross-platform attribute tag mapping technology. Furthermore, it utilized the Quaternion Spherical Linear Interpolation smooth camera movement algorithm to endow the system with immersive intelligent cruising capabilities, realizing an experience where "alarm equals localization, and localization equals visual confirmation." Coupled with the remote reset function featuring a timestamp anti-shake mechanism, the system successfully closed the loop on the full business link of perception, analysis, 3D mapping, and reverse control.

In summary, this research successfully established the full-stack technical pipeline from low-level hardware joint data acquisition and high-speed cloud routing to upper-level host 3D simulation. All functional metrics met or exceeded the anticipated design requirements, validating the high feasibility of deeply integrating digital twin technology with the IoT in the smart fire protection sector. It provides a technical solution and practical paradigm with significant reference value for the security digital transformation of universities and similar small-space, high-density buildings.

Acknowledgments

The work was supported by Sichuan Provincial Higher Education Talent Cultivation Quality and Teaching Reform Project (JG2024-0780, JG2024-0764), Graduate Education Teaching Reform and Practice Project (YJG202318, YJG202406, YJG202407) and Natural Science Foundation of Sichuan Province (2024ZYD0003, 2025ZYD0173), Graduate Education Reform and Practice Project (YJG2025 15).

References

[1] Xilong Q, Shengzong L, Sha F, et al. Design and implementation of wireless environment monitoring system based on STM32[J]. Scientific programming, 2021, 2021(1): 6070664.

[2] Kwon O H, Cho S M, Hwang S M. Design and implementation of fire detection system[C]. 2008 Advanced Software Engineering and Its Applications. IEEE, 2008: 233-236.

[3] Wang B, Zhu R, Luo H. Design and Implementation of STM32-based Intelligent Fire Response System for Fire Protection[J]. International Core Journal of Engineering, 2025, 11(7): 57-63.

[4] Baek J, Alhindi T J, Jeong Y S, et al. Intelligent multi-sensor detection system for monitoring indoor building fires[J]. IEEE Sensors Journal, 2021, 21(24): 27982-27992.

[5] Liu Z, Zhang A, Wang W. A framework for an indoor safety management system based on digital twin[J]. Sensors, 2020, 20(20): 5771.

[6] Handosa M, Gračanin D, Elmongui H G. Performance evaluation of MQTT-based internet of things systems[C]. 2017 Winter simulation conference (WSC). IEEE, 2017: 4544-4545.

[7] Longo E, Redondi A E C. Design and implementation of an advanced MQTT broker for distributed pub/sub scenarios[J]. Computer Networks, 2023, 224: 109601.

[8] Wang J, Ke T, Hou M, et al. The design of home fire monitoring system based on NB-IoT[J]. International Journal of Advanced Computer Science and Applications, 2022, 13(5).

[9] Zhang X, Jiang Y, Wu X, et al. AIoT-enabled digital twin system for smart tunnel fire safety management[J]. Developments in the Built Environment, 2024, 18: 100381.

[10] Moshood T D, Rotimi J O B, Shahzad W, et al. Infrastructure digital twin technology: A new paradigm for future construction industry[J]. Technology in Society, 2024, 77: 102519.

[11] Xie W, Zeng Y, Zhang X, et al. AIoT-powered building digital twin for smart firefighting and super real-time fire forecast[J]. Advanced Engineering Informatics, 2025, 65: 103117.

[12] Almatared M, Liu H, Abudayyeh O, et al. Digital-twin-based fire safety management framework for smart buildings[J]. Buildings, 2023, 14(1): 4.

[13] Chakravarthy R. An experimental study of IoT-Based topologies on MQTT protocol for agriculture intrusion detection[J]. Measurement: Sensors, 2022, 24: 100470.

[14] Udurume M, Hwang T, Uddin R, et al. Developing a fire monitoring system based on mqtt, esp-now, and a rem in industrial environments[J]. Applied Sciences, 2025, 15(2): 500.

[15] Gupta P K, Singh M K. AN IOT-ENABLED MULTI-SENSOR FRAMEWORK FOR FIRE DETECTION AND ALARM SYSTEMS: ENHANCING SAFETY THROUGH SECURE DATA ANALYTICS[J]. Scientific and practical cyber security journal, 2025.

[16] Jiang L, Shi J, Wang C, et al. Intelligent control of building fire protection system using digital twins and semantic web technologies[J]. Automation in construction, 2023, 147: 104728.

[17] Tao F, Qi Q, Wang L, et al. Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison[J]. Engineering, 2019, 5(4): 653-661.

[18] Abdullahi U I, Zhang W, Cao Y, et al. Integrating IoT technology for fire risk monitoring and assessment in residential building design[J]. Buildings, 2025, 15(8): 1346.

[19] Hua S U N. Internet of Things-Driven Safety and Efficiency in High-Risk Environments: Challenges, Applications, and Future Directions[J]. International Journal of Advanced Computer Science & Applications, 2025, 16(5).

[20] Abdullahi U I, Zhang W, Cao Y, et al. Integrating IoT technology for fire risk monitoring and assessment in residential building design[J]. Buildings, 2025, 15(8): 1346.

[21] Grieves M. Digital twin: manufacturing excellence through virtual factory replication[J]. White paper, 2014, 1(2014): 1-7.

[22] Saha N, Paul P, Ji K, et al. Performance evaluation framework of MQTT client libraries for IoT applications in manufacturing[J]. Manufacturing Letters, 2024, 41: 1237-1245.

[23] Mishra B, Mishra B, Kertesz A. Stress-testing MQTT brokers: A comparative analysis of performance measurements[J]. Energies, 2021, 14(18): 5817.

[24] Sun Q, Turkan Y. A BIM-based simulation framework for fire safety management and investigation of the critical factors affecting human evacuation performance[J]. Advanced Engineering Informatics, 2020, 44: 101093.

[25] Korhonen T, Hostikka S. Fire Dynamics Simulator with Evacuation: FDS+ Evac Technical Reference Guide and User's Guide[R]. VTT Working Papers 119, VTT Technical Research Center of Finland, 2009.

[26] Eastman C M. BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors[M]. John Wiley & Sons, 2011.

[27] McGrattan K, Hostikka S, McDermott R, et al. Fire dynamics simulator user's guide[J]. NIST special publication, 2013, 1019(6): 1-339.

[28] Dimyadi J A W, Spearpoint M, Amor R. Generating fire dynamics simulator geometrical input using an IFC-based building information model[J]. J. Inf. Technol. Constr., 2007, 12: 443-457.

[29] Guo Jr H, Magid E, Hsia K H, et al. Development of IoT module with AI function using STM32 chip[J]. Journal of Robotics, Networking and Artificial Life, 2021, 7(4): 253-257.

[30] Gourlis G, Kovacic I. A holistic digital twin simulation framework for industrial facilities: BIM-based data acquisition for building energy modeling[J]. Frontiers in Built Environment, 2022, 8: 918821.